OULU UNIVERSITY OF
APPLIED SCIENCES

Mai Thanh Tung

ANDROID APP TO SEARCH LOCAL VENUES WITH FOURSQUARE API

ANDROID APP TO SEARCH LOCAL VENUES WITH FOURSQUARE API

Mai Thanh Tung
Bachelor's Thesis
Spring 2018
Degree Programme in Information Technology
Oulu University of Applied Sciences

ABSTRACT

Oulu University of Applied Sciences
Degree programme in Information Technology

Author(s): Mai Thanh Tung
Title of Bachelor´s thesis: Android App to Search Local Venues with Foursquare API
Supervisor(s): Kari Laitinen
Term and year of completion: Spring 2018 Number of pages: 27

This Bachelor's Thesis is about the process of developing a simple REST API Android application. The purpose of the application to search venues that are near the user's current location.

The application was built by the author and the API was provided by Foursquare. The application was built using the knowledge that the author acquired from his degree programme and from his own research about Android development, as well as during his summer training with the Nordea digital team. The task was to build an application that retrieves data from Foursquare via their API. The development process is described in this paper. The application was built by using the Android Studio.

The result is that the application performs well and fulfils all the requirements. The design of the application could be improved. Also, it might be useful to add keyword hints for the users.

Keywords:

Android, Java, xml, Android development, mobile application, API, REST, client-side

CONTENTS

VOCABULARY

1. Java: It is one of the computer programming languages. Almost all the Android applications are developed with the Java language.
2. XML: It is a markup language that defines a set of rules for encoding documents. XML is used for designing layouts in Android.
3. API: stands for Application Programming Interface. It is a set of clearly defined methods of communication between various software components.
4. REST: stands for REpresentational State Transfer. The web service built with REST architectural style provides smooth interaction between computer systems on the Internet.
5. Client-side: It refers to actions that take place on the users' devices.

1 INTRODUCTION

1.1 App to search venues with Foursquare API

This thesis describes the development of an application that retrieves data from API. The application was built by the author only and the API was provided by Foursquare. The application was built using the knowledge that the author acquired from his degree, his summer training with the Nordea digital team and also his own research about Android development. The application was built by using the Android Studio.

1.2 What is Android and why Android

Android is a mobile operating system. At first it was developed by Android Inc. In 2005, it was bought by Google. Initially, Android was designed for mobile phones; afterwards, it evolved to be used for various devices such as televisions (Android TV), smart watches (Android wear), cars (Android Auto).

Since 2013, Android has been the best-selling OS on tablets and has been run on the vast majority of smartphones. In May 2017, Android reached two billion monthly active users. It also had the largest installed base compared to other operating systems, which means that Android applications had gained a large number of users. (1)

From the point of view of developers, the highly customizable nature of Android encouraged people to freely show their creativity. Android applications are written with Java programming language using the Android software development kit (SDK). Figure 1 below shows the history of the Android Operating System.

FIGURE 1. History of Android Operating System (1)

1.3 Foursquare

Foursquare is a local search and discovery service that can be used on mobile apps. It was created in 2008 and launched in 2009 by Dennis Crowley and Naveen Selvadurai. In July 2014, Foursquare featured a social networking layer that helps users share their locations with friends via “check-in”. In 2013, Foursquare reached 45 million registered users. Features of Foursquare include: Local search and recommendations, tips and expertise, and location detection. Also the one that the author of this thesis used for building the application is Foursquare API - a feature that lets third party applications use the location data of Foursquare. (2)

2 ANDROID DEVELOPMENT

2.1 Tools – Android Studio

Android studio is the official integrated development environment for Android. It was built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. The first stable build was released in December 2014 with version 1.0. The current stable version is 3.0 released in October 2017. (3)

Android Studio provides several features such as:

- Gradle-based build support (used in this thesis for build support)
- Lint tools to catch performance, usability, version compatibility, and other problems
- Android Virtual device to run and debug apps in Android Studio (used by author in this thesis' work for testing in virtual device)
- Android Device Monitor tool allows developer to monitor plugged devices (used by author in this thesis' work for installing and testing in a real device)

An example of Android Studio in Figure 2.

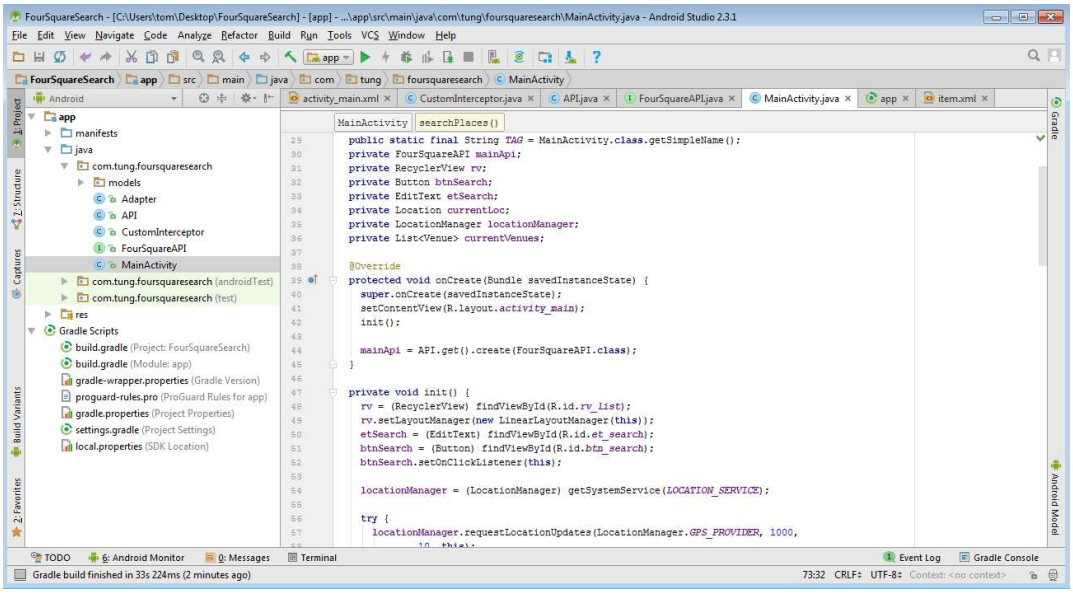

FIGURE 2. Android Studio

The Android studio also supports other languages than Java, such as Kotlin. Overall, the Android studio provides the fastest tool for building an app for every type of Android device.

2.2 Android application fundamentals

Each Android app lives in its own security sandbox, protected by some Android security features. Each application, by default, has access only to those components required to perform its functions and nothing more. This ensures the application cannot access to the parts of the system that are not given permission.

2.2.1 App components

App components are the essential building blocks of an app. They are entry points through which the system or user can enter the app. (4)

There are four different components:

- Activities: These are entry points interacting with the user represents a single screen with user interface. The application presented in this thesis has the screen to show list of the venues and also an input field to search. It is called MainActivity.
- Services: These are entry points for running the app in the background. It does not provide a user interface. For example, a music application is still playing songs even it is not shown on the screen.
- Broadcast receivers: These enable the system to deliver events to the app even when the app is not running. For example, broadcast can announce to applications that the screen is turned off or the battery is low.
- Content providers: These manage a shared set of app data that can be stored in the file system, in SQLite database... For example, some applications like Viber or WhatsApp require access to the phone's data of contacts so that it reads and writes information about contacts in their applications.

To activate components:

- Activities can be started by passing an Intent to startActivity() or startActivityforResult().

- Services can be activated by passing an Intent to StartService(). Also you can bind to the service by passing an Intent to bindService().
- The broadcast is initiated by passing an Intent to methods such as sendBroadcast(), sendOrderedBroadcast() or sendStickyBroadcast().
- A query is performed to a content provider by calling query() on a ContentResolver.

2.2.2 Manifest file

The manifest file declares all of the application's components so that the system can acknowledge the components exist.

The manifest file can:

- Identify the User permission that the app requires.
- Declare the minimum API level that the app requires.
- Declare the hardware and software features used or required by the app.
- Declare the API libraries the app needs to be linked with.

2.2.3 App resources

App resources are all the resources that an application needs. Apart from code, these resources can be images, audio files and anything related to the UI of the application such as style, colors, and the layout of the activity's user interfaces.

2.3 APK file

The Android Package Kit (APK) is used by Android for distribution and installation of mobile app and middleware. Once an application is completed, it can be compiled into an APK file in order to publish it so that others can download and install it via Google Play Store or some website.

3 LOCAL SEARCH APPLICATION

In this part, it is explained how to make a simple Android application with REST API.

3.1 Introduction

The application will provide users with a searching feature to look for venues that are around the current location of them. The result will be based on what the users type in the search box and it will be triggered when they type a character.

Figure 3 shows the structure of the application. It includes 3 main parts:

- Models : Handling and storing the data that the application receives from Foursquare
- Controller : Handling the sending request to Foursquare server, user activity and display results
- View : UI code for showing application on screen

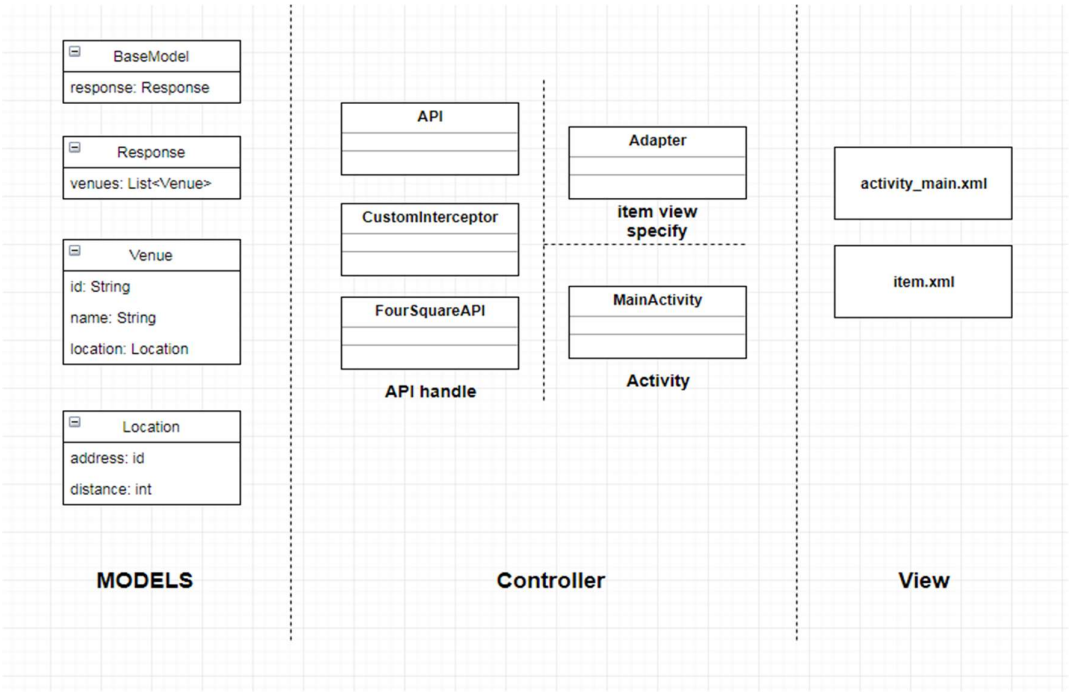

FIGURE 3. Application structure

3.2 Preparation

The following steps have to be taken to start the work:

- Foursquare: First register a developer account in the Foursquare (https://developer.foursquare.com/). Then, create a new Foursquare app to obtain the Client ID and Client Secret key.
- Android studio: Create a new project.
- Prepare test phones or install Android Virtual device of Android studio for testing purpose.

3.3 Developing

3.3.1 Data structure - create models

Figure 4 shows what kind of data can be received from Foursquare server.

FIGURE 4. Example of response data from Foursquare

Based on the response structure, we make the Models that will represent the underlying, logical structure of the data. For this application, we only get the address, distance, and the name of location.

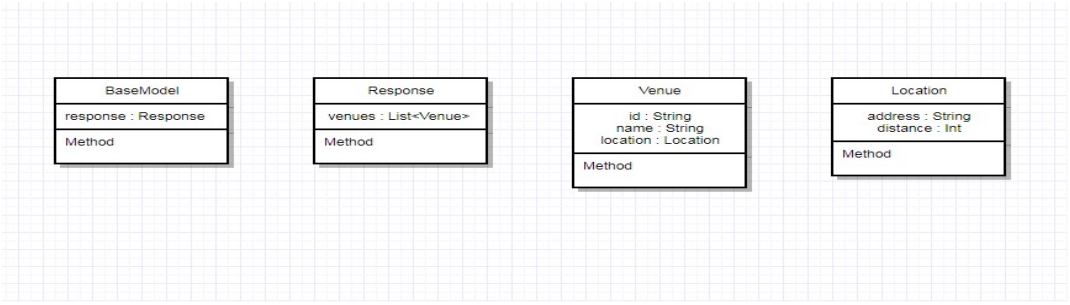

FIGURE 5. Models structure

Figure 6 shows the data structure of the application and what it looks like in the Android Studio. There is a folder called Models that contains all the classes that represent Json data. It has to be declared because Java could not read the Json response.

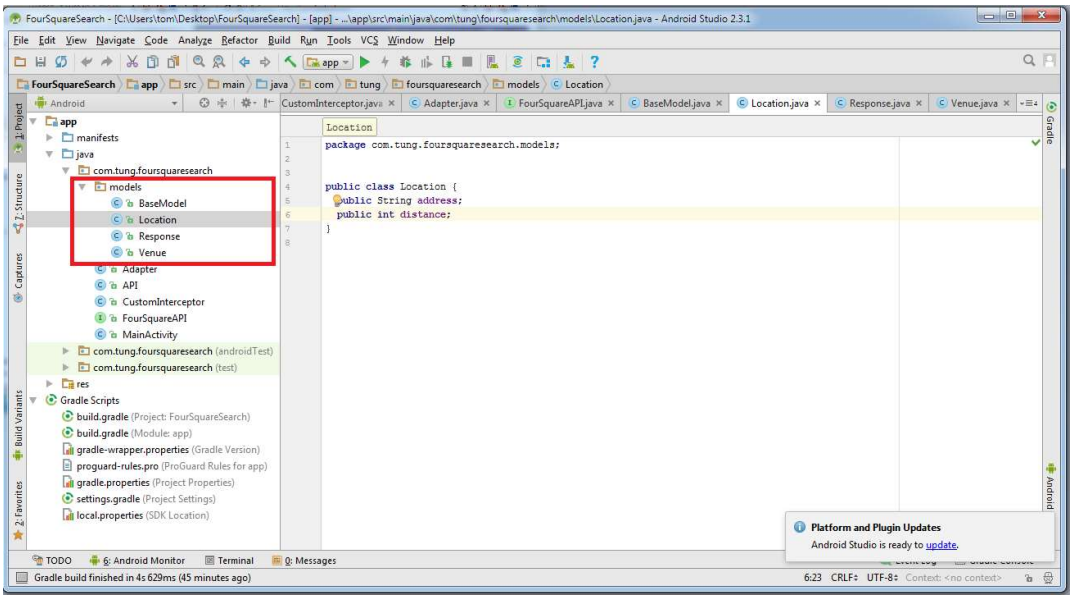

FIGURE 6. Models in Android studio

3.3.2 API class

A class containing the base URL of API is created and the Retrofit with OkHttpClient that contains our Custom Interceptor is built. Firstly, the base URL being used is declared, then the own Custom Interceptor is declared and added to the OkHttpClient. After that, the Retrofit Builder handles the client and url.

Figure 7 shows how the code looks like.

```
package com.tung.foursquaresearch;

import okhttp3.OkHttpClient;
import retrofit2.Retrofit;
import retrofit2.adapter.rxjava.RxJavaCallAdapterFactory;
import retrofit2.converter.gson.GsonConverterFactory;
import rx.schedulers.Schedulers;

public class API {
  public static final String BASE_URL = "https://api.foursquare.com/v2/";

  static CustomInterceptor interceptor = new CustomInterceptor();

  static OkHttpClient okHttpClient = new OkHttpClient.Builder()
          .addInterceptor(interceptor)
          .build();

  static RxJavaCallAdapterFactory rxAdapter = RxJavaCallAdapterFactory.createWithScheduler(Schedulers.io());

  static Retrofit retrofit = new Retrofit.Builder()
          .client(okHttpClient)
          .baseUrl(BASE_URL)
          .addConverterFactory(GsonConverterFactory.create())
          .addCallAdapterFactory(rxAdapter)
          .build();

  public static Retrofit get() { return retrofit; }
}
```

FIGURE 7. API class

3.3.3 Custom Interceptor

After the developer's account is created in Foursquare and a new app is declared, the CLIENT_ID and CLIENT_SECRET is received. This step is to sign the key to specify the application with the Foursquare server. Every time the application is used, it will automatically send requests to the Foursquare server. The server includes the key to let Foursquare know which application is requesting the data. Figure 8 shows the CustomInterceptor class.

```
package com.tung.foursquaresearch;

import java.io.IOException;

import okhttp3.HttpUrl;
import okhttp3.Interceptor;
import okhttp3.Request;
import okhttp3.Response;

public class CustomInterceptor implements Interceptor {
  @Override
  public Response intercept(Chain chain) throws IOException {
    Request original = chain.request();
    HttpUrl originalHttpUrl = original.url();
    HttpUrl url = originalHttpUrl.newBuilder()
            .addQueryParameter("insecure", "cool")
            .addQueryParameter("client_id", "CYEMKDM4OLTP5PHMOFVUJJAMWT5CH5G1JBCYREATW21XLLSZ")
            .addQueryParameter("client_secret", "GYNP4URASNYRNRGXR5UEN2TGTKJHXY5FGSAXTIHXEUG1GYM2")
            .addQueryParameter("v", "20171026")
            .build();
    Request.Builder requestBuilder = original.newBuilder()
            .url(url);
    Request req = requestBuilder.build();
    return chain.proceed(req);
  }
}
```

FIGURE 8. Custom interceptor class

3.3.4 Making Foursquare API Interface class

Figure 9 is an interface class which defines exactly which Foursquare API will be used.

```
package com.tung.foursquaresearch;

import com.tung.foursquaresearch.models.BaseModel;

import retrofit2.http.GET;
import retrofit2.http.Query;
import rx.Observable;

public interface FourSquareAPI {

  @GET("venues/search/")
  Observable<BaseModel> venueSearchLL(@Query("ll") String latlon);

}
```

FIGURE 9. FoursquareAPI Interface class

3.3.5 Adapter class and item.xml

In this thesis, as the RecyclerView has been used, the Adapter class has to be created to customize what will show in the item of RecyclerView and item.xml file to create the layout for the item interface. Figure 10 shows what the UI code of the Main Activity looks like.

In Figure 11 the data in one item of the Recycle view shows: name, address and distance.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:orientation="vertical"
              android:layout_width="match_parent"
              android:paddingTop="6dp"
              android:paddingBottom="6dp"
              android:paddingLeft="6dp"
              android:layout_height="wrap_content">

    <TextView
        android:textStyle="bold"
        android:textSize="18sp"
        android:layout_marginBottom="4dp"
        android:id="@+id/tv_name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>

    <TextView
        android:textSize="16sp"
        android:id="@+id/tv_address"
        android:layout_marginBottom="2dp"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>

    <TextView
        android:textSize="16sp"
        android:id="@+id/tv_distance"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>

    <View
        android:background="#000"
        android:layout_width="match_parent"
        android:layout_height="1dp"/>

</LinearLayout>
```

FIGURE 10. Item.xml file

```
package com.tung.foursquaresearch;

import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import com.tung.foursquaresearch.models.Venue;

import java.util.List;

public class Adapter extends RecyclerView.Adapter<Adapter.ViewHolder> {
  private List<Venue> venues;

  public Adapter(List<Venue> venues) { this.venues = venues; }

  @Override
  public Adapter.ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
    View v = LayoutInflater.from(parent.getContext())
            .inflate(R.layout.item, parent, false);

    return new ViewHolder(v);
  }

  @Override
  public void onBindViewHolder(ViewHolder holder, int position) {
    holder.tvName.setText(venues.get(position).name);
    holder.tvAddress.setText("Address: " + venues.get(position).location.address);
    holder.tvDistance.setText("Distance: " + String.valueOf(venues.get(position).location.distance));
  }

  @Override
  public int getItemCount() { return venues == null ? 0 : venues.size(); }

  public static class ViewHolder extends RecyclerView.ViewHolder {
    public TextView tvName;
    public TextView tvAddress;
    public TextView tvDistance;

    public ViewHolder(View itemView) {
      super(itemView);
      tvName = (TextView) itemView.findViewById(R.id.tv_name);
      tvAddress = (TextView) itemView.findViewById(R.id.tv_address);
      tvDistance = (TextView) itemView.findViewById(R.id.tv_distance);
    }
  }
}
```

FIGURE 11. Adapter class

3.3.6 Main activity

Before going into details of the MainActivity of the application, how an Activity works and its life cycle is explained in the following figures and facts. The Activity knows what the current state is: system is created, stopped, resumed or destroyed.

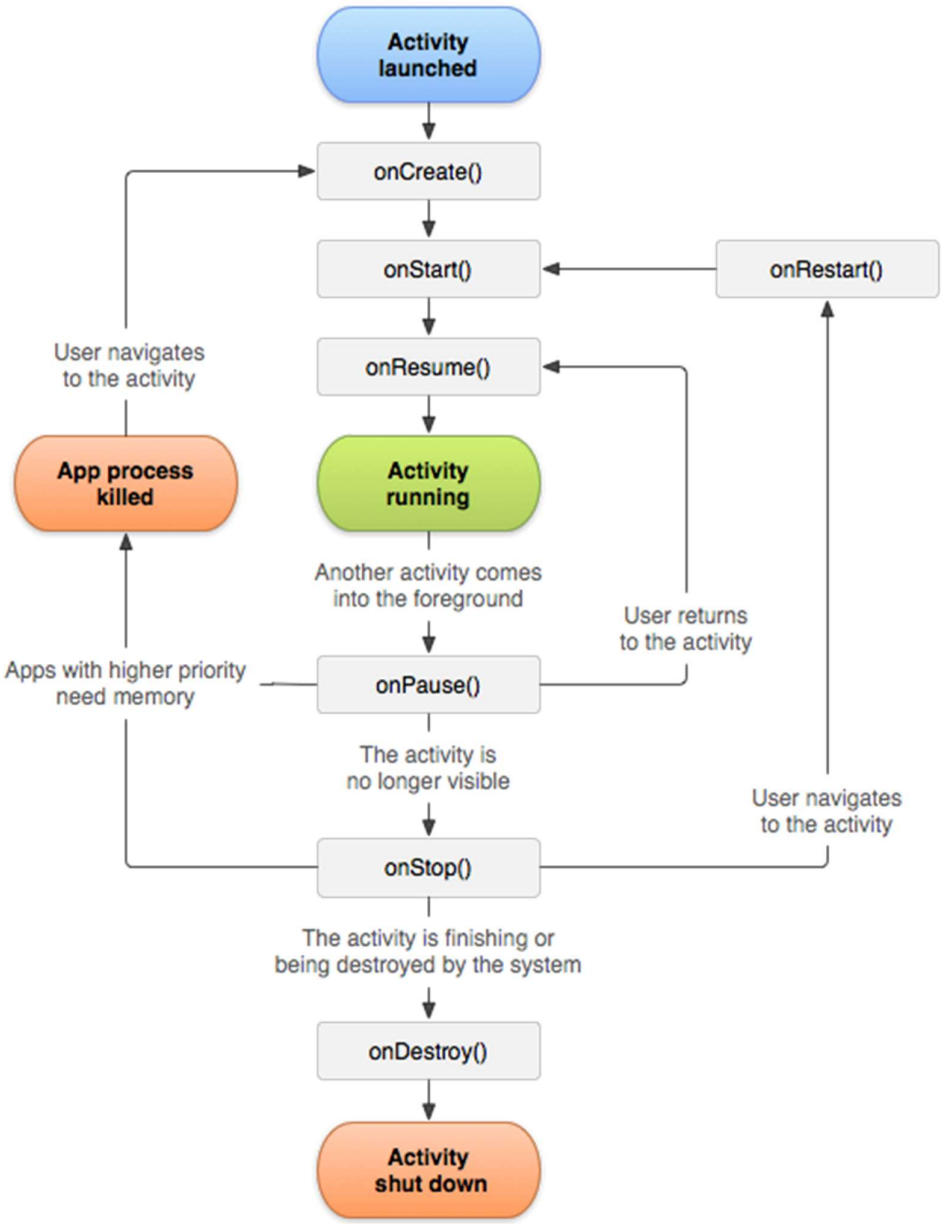

FIGURE 12. Activity Lifecycle (5)

As Figure 12 describes, the entire lifetime of an activity happens between onCreate() and onDestroy(). The activity is visible between onStart() and onStop(). The foreground lifetime of the activity happening when the user interacts with the application is between onResume() and onPause().

In the thesis, an application is created which can be used to search and provide results. Therefore, the layout has an EditText to let the users fill out the keywords to search, a Button to trigger the search, and a View to show the results. In this thesis, the RecyclerView has been used. The codes are shown in Figure 13.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context="com.howkteam.foursquaresearch.MainActivity">

    <EditText
        android:id="@+id/et_search"
        android:layout_width="match_parent"
        android:maxLines="1"
        android:layout_height="wrap_content"/>

    <Button
        android:id="@+id/btn_search"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Search"/>

    <android.support.v7.widget.RecyclerView
        android:id="@+id/rv_list"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>

</LinearLayout>
```

FIGURE 13. Activity_main.xml file

As mentioned above, the main activity begins with onCreate(). Figure 14 shows the function on Create() of Mainactivity.java

```
@Override
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);
  mainApi = API.get().create(FourSquareAPI.class);
  init();
}

private void init() {
  rv = (RecyclerView) findViewById(R.id.rv_list);
  rv.setLayoutManager(new LinearLayoutManager(this));
  etSearch = (EditText) findViewById(R.id.et_search);
  btnSearch = (Button) findViewById(R.id.btn_search);
  btnSearch.setOnClickListener(this);

  locationManager = (LocationManager) getSystemService(LOCATION_SERVICE);
  currentLoc = locationManager.getLastKnownLocation(LocationManager.GPS_PROVIDER);
  if (currentLoc != null) {
    updatePlaces(currentLoc);
  }

  try {
    locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 1000,
            10, this);
  } catch (Exception e) {
    Log.e(TAG, "Error", e);
  }

  etSearch.addTextChangedListener(new TextWatcher() {
    @Override
    public void beforeTextChanged(CharSequence s, int start, int count, int after) {

    }

    @Override
    public void onTextChanged(CharSequence s, int start, int before, int count) {
      searchPlaces(s.toString());
    }

    @Override
    public void afterTextChanged(Editable s) {

    }
  });
}
```

FIGURE 14. MainActivity.java

In onCreate():

- Set the view that will be used, in this case is activity_main.
- Declare the API that will be used.
- Set all the content in the view (the recyclerView, the button search and the editText field)

- Get the current location provided by the phone using GPS_PROVIDER.
- Try to update the location after every second or 10 meters away from the last location.
- Show the venues around the current location with the updatePlaces function.
- Search for venues when users type in editText Search field by searchPlaces function.

```
private void updatePlaces(Location loc) {
  String ll = String.valueOf(loc.getLatitude() + "," + String.valueOf(loc.getLongitude()));
  mainApi.venueSearchLL(ll)
          .delay(1, TimeUnit.SECONDS)
          .observeOn(AndroidSchedulers.mainThread())
          .subscribeOn(Schedulers.io())
          .subscribe(new Subscriber<BaseModel>() {
            @Override
            public void onCompleted() { Log.e(TAG, "Completed"); }

            @Override
            public void onError(Throwable e) { Log.e(TAG, "Error", e); }

            @Override
            public void onNext(BaseModel baseModel) {
              currentVenues = baseModel.response.venues;
              rv.setAdapter(new Adapter(currentVenues));
              rv.getAdapter().notifyDataSetChanged();
            }
          });
}
```

FIGURE 15. updatePlaces function

Figure 15 shows the updatePlaces function where we get the current Location, then use it to get to the venues with FoursquareAPI. The Foursquare server returns a baseModel result. All the venues are gotten and put into the Adapter of RecyclerView.

```
private void searchPlaces(String keyword) {
  if (TextUtils.isEmpty(keyword)) {
    rv.setAdapter(new Adapter(currentVenues));
  }
  if (currentVenues == null) return;
  List<Venue> newTempList = new ArrayList<>();
  for (Venue venue : currentVenues) {
    if (venue.name.toLowerCase().contains(keyword.toLowerCase())) {
      newTempList.add(venue);
      Log.e(TAG, venue.name);
    }
  }

  Log.e(TAG, "Length" + String.valueOf(currentVenues.size()));
  rv.setAdapter(new Adapter(newTempList));
}
```

FIGURE 16. searchPlaces function

In the searchPlaces function shown in Figure 16, the keywords that the user typed in are acknowledged. First, the keywords are checked to see if they are empty or not, then all venues are set into RecyclerView. Next, the venues are checked whether they are available, if not, the application shows nothing. Finally, the venues are examined to see whether in the list of venues, there are venues contain the keywords that user typed in, then they are added to a new list (in this case is the newTempList). After all this is done, that new list is put into the RecyclerView.

3.3.7 Manifest file

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
          package="com.tung.foursquaresearch">

    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="FourSquareSearch"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name="com.tung.foursquaresearch.MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>

                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
    </application>

</manifest>
```

FIGURE 17. Manifest file

Figure 17 shows the code for the manifest file. In here, the user's permission is asked for using the Internet service and location service. Also in here, the MainActivity will be declared for the launcher of the application.

3.3.8 Testing

It is normal for humans to make mistakes. Some mistakes are not important, but some are dangerous and expensive. For this reason, software needs to be tested. The Android studio offers supports for developers in testing, such as Debug feature and Android monitor, which are shown in Figure 18. When using an Android device or simulator to run the application or debug the application, the Android monitor will point out everything that is happening when the device runs the application such as bugs, coding logs and performance issues (6). There are also other ways of testing such as Unit testing or Automated testing.

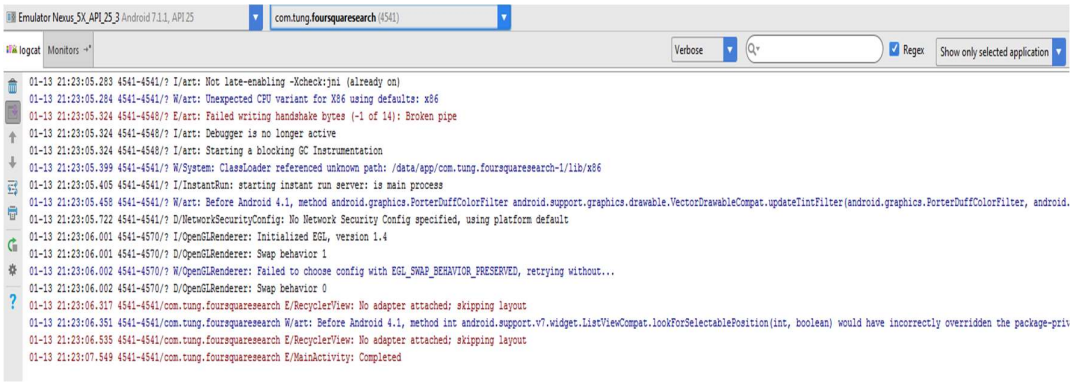

FIGURE 18. Android monitor

3.3.9 Publishing

After all the requirements have been met and all the functions have been tested, the application can be published. An APK file is generated from the application by Android Studio's Generate Signed APK feature. The APK file can be directly put and installed in an Android device, or it can be published to the Google Play store, which is the app store for Android devices. Android requires all the applications to be digitally signed with a certificate before they can be installed. Therefore, in order to publish the application via Google Play store, developers must generate a signed release APK. Figure 19 shows one step of generating.

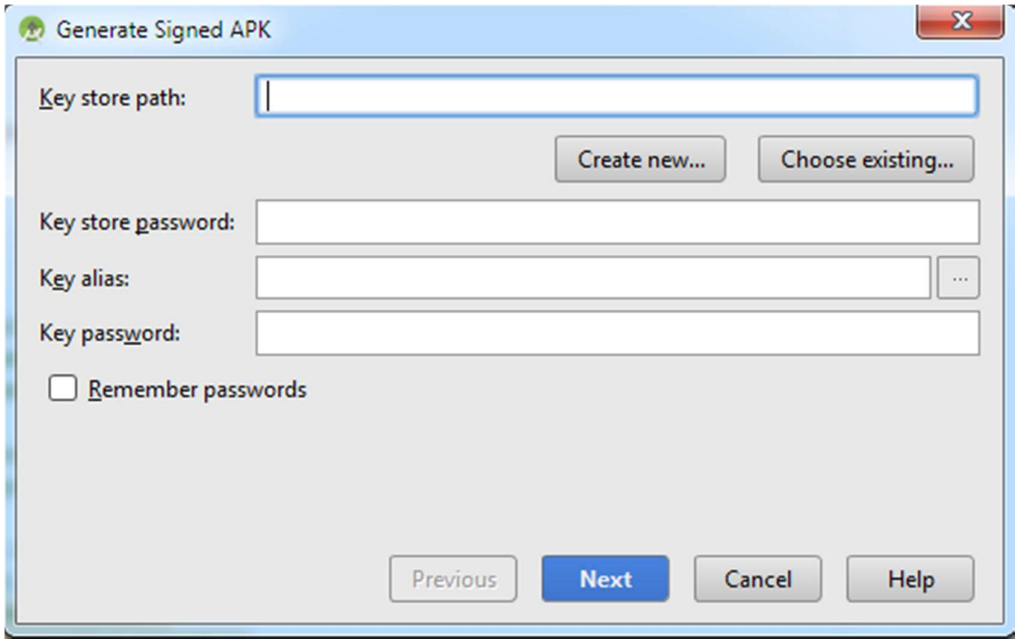

FIGURE 19. Generate Signed APK file

The key store is like an ID of the application, which means that if developers use a different key store in the exactly same application, Android still considers it as a different application. When publishing the app to Google Play store with a different keystore, the Android system will delete the older application and install the new one instead of updating it.

3.4 The result

When the user opens the application, the launcher screen will be displayed as shown in Figure 20. It will show all the venues around the current location of the user. Figure 21 shows that when users start to type in something to the search input field text box, the venues will get sorted to show nearly exact results compared to what users typed in. And the final result can be seen in Figure 22.

FIGURE 20. First screen

FIGURE 21. User starting to type

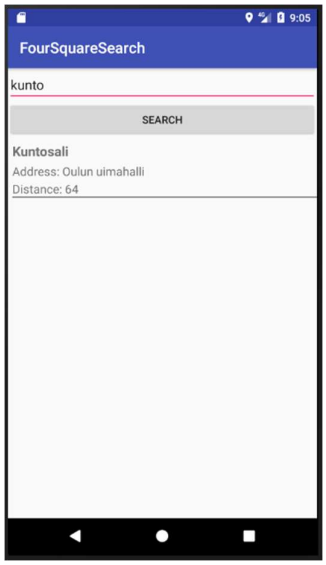

FIGURE 22. Final result

4 CONCLUSION

Learning and building an Android application can be both easy and hard at the same time due to all the figures and facts mentioned throughout the thesis. It takes time, and requires the developers to deeply understand what they are dealing with.

The main purpose of this thesis is to develop a simple REST API Android application. All the information leads to the result that the application that searches venues near users' current location is built. Throughout the process of making an application, the author has acquired more knowledge in Java programming, Android development as well as how to publish an application to Google Play store.

However, while developing an application for mobile devices, the problems lie in the mobile device itself. To be more precise, bandwidth, hard drive space and memory, or battery power are all the drawbacks. What is more, there are too many Android OS versions existing. This brings out the situation where almost modern Android phones work much smoothly compared to the old-versioned ones. Actually, these problems are general issues that no one has the ability to fix them completely, but developers do have the ability to restrict some parts of them such as coding for the mobile devices to use less batteries, or to spend less hard drive spaces and memories.

Long story short, the principle objectives of this thesis, which are to develop an Android application with REST API, have all been accomplished.

REFERENCES

1. Wikipedia. Android (operating system). Date of Retrieval 20.12.2017, https://en.wikipedia.org/wiki/Android_(operating_system)

2. Wikipedia. Foursquare. Date of Retrieval 20.12.2017, https://en.wikipedia.org/wiki/Foursquare

3. Wikipedia. Android Studio. Date of Retrieval 20.12.2017, https://en.wikipedia.org/wiki/Android_Studio

4. Android's developer document. Application Fundamentals. Date of Retrieval 20.12.2017, https://developer.android.com/guide/components/fundamentals.html

5. Android's developer document. Activity. Date of Retrieval 20.12.2017, https://developer.android.com/reference/android/app/Activity.html

6. React native community. Generating signed APK. Date of Retrieval 20.12.2017, https://facebook.github.io/react-native/docs/signed-apk-android.html